PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-161933

(43)Date of publication of application : 19.06.1998

(51)Int.Cl. G06F 12/14
G06F 9/46

(21)Application number : 09-254505

(22)Date of filing : 19.09.1997

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(72)Inventor : DONALD FRED OORT
ERNEST SCOTT BENDER
MICHAEL GARY SPIEGEL

(30)Priority

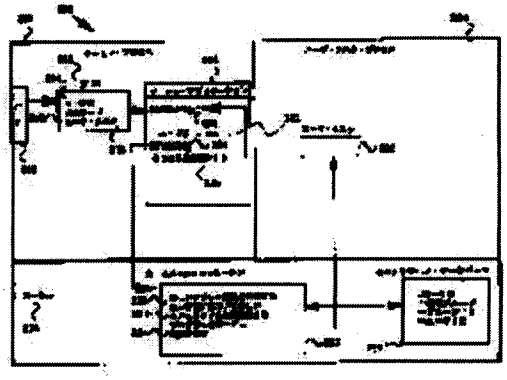
Priority number : 96 721145 Priority date : 26.09.1996 Priority country : US

(54) METHOD AND DEVICE FOR USER TASK EXECUTION IN PROPER SECURITY ENVIRONMENT IN CLIENT SERVER SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To execute a task for a user by executing a user task specified in a newly prepared address space.

SOLUTION: Demon application 208 prepares new environment after verifying a user ID 214 and authenticating a password 216. A kernel span routine 226 in a kernel layer 204 acquires control and generates a new user task layer or address pace 206. A POSIX permission set value is inquired of a security data base 232 in the kernel layer 204 and obtained. The new address space 206 is initialized directly by using a group ID made to correspond to a user name 214 in the data base 232 and the user ID and group ID of the new address pace 206 set equal to the user ID. A program image 238 corresponding to the user task 218 is loaded in the address pace 206 and the control is passed to the program image to execute the user task.



* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1]. A demand specifies a user's identity and a server system has an operating kernel. In a server system with which a daemon process supervises said demand from said user who asks for execution of a specified user task, Are the method of performing said task instead of said user using suitable security environment for said user, and if the (a) daemon process receives said demand from said user, [1] A step which sets up an environment variable according to said identity specified by said demand, [2] A step which takes out a system call to said operating system kernel, and performs said specified user task in a new address space, (b) If said operating system kernel receives said system call from said daemon process, [1] A step which creates an address space new for said specified user task, [2] a step which creates security environment according to said environment variable for said specified user task, and [3] -- a method containing a step which starts said specified user task in said new address space.
[Claim 2]A method according to claim 1, wherein said operating system kernel is a POSIX conformity operating system kernel.
[Claim 3]A method according to claim 1, wherein said system call is a spawn() system call.
[Claim 4]A method according to claim 1 which said user has a user name and is characterized by said identity containing said user name.
[Claim 5]A method according to claim 4, wherein said environment variable is set up equally to said user name.
[Claim 6]Said new address space and said user name have the user ID related with it, respectively, A method according to claim 5, wherein said Step (b) [2] which creates said security environment sets up equally to user ID of a user name specified by said environment variable user ID of said new user address space.
[Claim 7]A method of judging user ID of said user name by accessing a security database

according to claim 6.
[Claim 8]Said new address space and said user name have the group ID related with it, respectively, A method according to claim 5, wherein said Step (b) [2] which creates said security environment sets up group ID of said new address space equally to group ID of a user name specified by said environment variable.
[Claim 9]Said server system with which a daemon process supervises said demand from said user who asks for execution of a specified user task characterized by comprising the following whose server system a demand specifies a user's identity and has an operating kernel.
Are a device which performs said task instead of said user using suitable security environment for said user, it is related with the (a) aforementioned daemon process, and a receipt of said demand from said user is answered, [1] A means to set up an environment variable according to said identity specified as said demand, to take out a system call to the [2] aforementioned operating system kernel, and to perform said specified user task in a new address space.
(b) It is related with said operating system kernel, and a receipt of said system call from said daemon process is answered, [1] creating an address space new for said specified user task -- [2] -- creating security environment according to said environment variable for said specified user task -- [3] -- a means to start said specified user task in said new address space.

[Claim 10]The device according to claim 9, wherein said operating system kernel is a POSIX conformity operating system kernel.
[Claim 11]The device according to claim 9, wherein said system call is a spawn() system call.
[Claim 12]The device according to claim 9 which said user has a user name and is characterized by said identity containing said user name.
[Claim 13]The device according to claim 12, wherein said environment variable is set up equally to said user name.
[Claim 14]Said new address space and said user name have the user ID related with it, respectively, The device according to claim 13 containing a means by which said means (b) [2] which creates said security environment sets up user ID of said new address space equally to user ID of a user name specified by said environment variable.
[Claim 15]The device according to claim 14 judging user ID of said user name by accessing a security database.
[Claim 16]Said address space and said user name have the group ID related with it, respectively, The device according to claim 13, wherein said Step (b) [2] which creates said security environment sets up group ID of said new address space equally to group ID of a user name specified by said environment variable.
[Claim 17]. A demand specifies a user's identity and a server system has an operating system kernel. In said server system with which a daemon process supervises said demand from said

user who asks for execution of a specified user task, It is the machine-readable program storage which carries out to concreteness a program which can be executed with machinery which performs a method step which performs a task instead of said user using suitable security environment for said user, If the (a) aforementioned daemon process receives said demand from said user, said method step, [1] A step which sets up an environment variable according to said identity specified as said demand, [2] A step which publishes a system call to said operating system, and performs said specified user task in a new address space, (b) A step which creates an address space where it is new for a user task by which [1] aforementioned specification was carried out when said operating system kernel receives said system call from said daemon process, [2] a step which creates security environment according to said environment variable for said specified user task, and [3] -- program storage containing a step which starts said specified user task in said new address space.

[Claim 18]The program storage according to claim 17, wherein said operating system kernel is a POSIX conformity operating system kernel.

[Claim 19]The program storage according to claim 17, wherein said system call is a spawn() system call.

[Claim 20]The program storage according to claim 17 which said user has user ID and is characterized by said identity containing said user ID.

[Claim 21]The program storage according to claim 20, wherein said environment variable is set up equally to said user ID.

[Claim 22]Said new address space and said user name have the user ID related with it, respectively, The program storage according to claim 21 containing a step to which said Step (b) [2] which creates said security environment sets user ID of said new address space equally to user ID of a user name specified by said environment variable.

[Claim 23]The program storage according to claim 22 judging said user ID of said user name by accessing a security database.

[Claim 24]Said new address space and said user name have the group ID related with it, respectively, The program storage according to claim 21 containing a step to which said Step (b) [2] which creates said security environment sets group ID of said new address space equally to group ID of a user name specified by said environment variable.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention]This invention relates to the method of creating a new operating unit by the POSIX environment using a new user identity and a suitable privilege.
[0002]
[Description of the Prior Art](Work is distributed by two or more machinery by which interconnection was carried out) A distributed computing system is built based on a client/server model in many cases. In this model, a client process (or only "client") receives a server process (or only "server"), In the case of a print server, printing of a file, and in the case of a file server, the demand for which it asks so that specified services, such as execution of application, may be performed in the case of extraction of a file, memory, or an application server is advanced. Although it can exist on the physical machinery with same client process and server process, as for both, existing on different machinery is common, and it is so also in the following explanation. A server is used also in the network of a wide area from those which is local area networks (LAN) etc., such as others and the Internet. One class of the Internet server treated especially on these specifications is a server which provides service for World Wide Web. World Wide Web is a meeting of the Internet site which provides graphical contents ("web page") and serves others according to a HyperText Transfer Protocol (HTTP).
[0003]Servers, such as a server on World Wide Web (or only "web"), function with an operating system, and an operating system manages a system resource on the machinery with which a server exists, and offers base system service. These operating systems are a UNIX operating system or an operating system of a UNIX base in many cases. Various efforts to define the set of the common service provided by such a system by rapid increase of various UNIX operating systems have been made. Such one efforts are the 1003.IEEE POSIX1 first specification to be announced in 1988, and a supplement of 1003.1 d IEEE POSIX etc. (it

collects below and referred to as "POSIX").
[0004]Access to the file by a user, etc. is controlled by a POSIX conformity system by associating the user ID and group ID which specify a user's identity as each process and each file. The three triplet fields are also associated and each field changes from the permission bit of the triplet called a rwx bit to each file. These three fields define access permitted to the owner of a file, other users in an owner's group, and other users that do not go into the owner's group, respectively. Within three bit fields each, it specifies whether r bit can perform reading of a file, and specifies whether w bit can do the writing of a file, and it is specified whether x bit can perform execution of a file. It is judged whether when a process requires access to a file, the permission bit specified as a process, the user ID of a file, and group ID for the file can certainly be inspected, and access which the user who is performing the process demands can be performed. This access control procedure is common knowledge in this technical field. "Stevens (Stevens) W.R. UNIX. It is indicated to reference works, such as Network Programming" (1990) and A.S. Tanenbaum's (Tanenbaum) "Modern OperatingSystem" (1992).

[0005]In order to realize such access control in a server environment, a POSIX conformity operating system, The capability of the server which can create the new security environment which can access 1 set of resources in which a new operating unit differs from its new operating unit using a new user identity must be supported. The work of this type has been conventionally done by the background program called a "demon" within a server system. Generally, a demon processes a demand instead of the user who has a subset of a demon's authority, and performs a task. In order to maintain the security of a system, a task must be performed under a new operating unit using the user's authority. Creation of these new operating units is treated using the POSIX service to which functions, such as fork(), setuid(), setgid(), and exec(), and others relate.
[0006]A demon's example is a Web server on a POSIX system. Such a Web server is a program which supervises that a user hands in a port the demand which takes out a document from a POSIX file system. In order to perform this, the Web server must verify a user name and a password first. Next, a Web server creates a new address space using a fork() function, and provides the environment where it is separate for the demanded user task. If it goes into a new address space, a Web server will create right security environment using various POSIX functions, such as getgroups() and setgroups(), After setting up right supplementary group ID, user ID must be set to right group ID using setgid() and setuid(). A supplementary group, group ID, and user ID form the foundation of POSIX permission. Finally a Web server performs the shell script which performs exec() of shell and gains the demanded document. Seeing on the whole, this is the long and complicated process of requiring a lot of processing overheads.

[0007]
[Problem(s) to be Solved by the Invention]spawn() service is a POSIX function which combines fork() service and exec() service and is made one call (indicated to 1003.1 d IEEE POSIX). Therefore, POSIX application, Instead of performing a new program image, after forking a new address space, the new program can only be created in a different address space, and the overhead which copies the address space of a requestor side can be saved. However, the problem in this case is that there is no method of changing a user identity, before a new program image gains control at present, without changing the demon's itself security environment. Therefore, if it is going to create a shell script in order to gain the document in which the demon was demanded, Access to a POSIX file system will be performed under a demon's user identity, and POSIX permission of the user who is demanding the document by it may be replaced. Since spawn() cannot be used, demon application must be provided with the excessive troublesome software performed by a user task process layer as mentioned above.
[0008]
[Means for Solving the Problem]Generally, in a server system with which a daemon process supervises a demand from a user who asks to perform a specified user task, this invention relates to a method and a device which perform a task instead of a user using suitable security environment for a user. If a demand is received from a user, a daemon process sets up an environment variable according to a user identity (for example, user name) specified by the demand, and receives an operating system kernel, A system call which performs a specified user task in a new address space is taken out. An operating system kernel can be made into a POSIX conformity kernel, and a system call can be made into a spawn() system call in that case. If a system call is received from a daemon process, an operating system kernel, An address space new for a specified user task is created, new security environment is created according to an environment variable for a specified user task, and a user task specified in a new address space is performed.
[0009]Specifically, according to this invention, the above-mentioned fault of a Prior art is canceled by creating a new environment variable (USERNAME). If this environment variable is specified, a called program will gain control by a spawn() function using right POSIX permission. In order to realize this, after changing a spawn() function, and a spawn() function's recognizing this new environment variable and verifying that it is an effective user name, initial creation of an address space new for a specified user name is made to be performed. If it performs using a new user identity, the remaining POSIX permission will come to hand from an entity of the user included in a security data base. If initial setting of a new address space and a task is completed, a spawn routine will start the new program image by the present security environment.
[0010]This invention provides a method of creating security environment for a user task

process without the necessity of adding change to daemon process environment. It is necessary to perform no portion of demon application in a user task process (if that is right, before passing control to a user task program image, change of user task process permission will be needed). It can ** that this invention replaces use of conventional fork() and exec() by use of a spawn() function, and simplification and performance improvement which a spawn() function brings about intrinsically by it are realized.

[0011]

[Embodiment of the Invention]Drawing 1 is an example of the computer systems 100 (physical machinery is included) incorporating the demon of the conventional embodiment, and the relation between various software layers which perform the task demanded for the specific user is shown. A demon process layer or the address space 102, an operating system (OS), a kernel layer or the address space 104, and the user task process layer or the address space 106 created by carrying out like the after-mentioned is included in these layers. Physical machinery can be made into IBM S/390 processors, such as S / 390 parallel enterprise server, for example, and the kernel layer 104 can be made into the IBM OS/390 operating system which has a POSIX conformity OpenEdition component.

[0012]There is the demon application 108 including the software which supervises the port or the communication line 110 which operated instead of the user and was combined with the remote client (not shown) in the demon process layer 102. The demon application 108 accepts the demand 112 containing the identifier 118 which specifies the user name 114, the password 116, and a task as an input via the port 110 (Step 120).

[0013]The user name (or login name) 114 is an alphanumeric-characters sequence used in order that it may be related peculiar to a user and a user may make the system 100 identify oneself. The integer user ID related with the record 136 of each user name 114 in the security database 134 which accompanies the system 100 peculiar to a user name, A user's besides the integer group ID associated peculiar to a user name and the list of supplementary groups related with the user name password and other relevant information are memorized. In order to attest a user, the demon application 108 accesses the record 136 corresponding to the specified user name 114 in the security database 134, and investigates the password included in the record. There is the record 136 of the specified user name 114, and when the password in a record is in agreement with the password 118 attached to the demand, it is attested with a claimant being a user which the claimant is calling itself.

[0014]The demon application 108 verifies the user name 114, and after it attests the password 116, it must prepare new environment so that the new program image which performs the demanded task may be performed. Therefore, the demon application 108 takes out a fork() system call to the kernel layer 104, and creates a new process (Step 122).

[0015]A fork() system call carries out the trigger of the kernel fork routine 124 in the kernel

layer 104, and makes control gain. If control is gained, the fork routine 124 will create and initialize the new address space 106 (Step 126), and will copy a memory attribute, security attributes, and a processes run attribute to the user task process layer 106 from the demon layer 102 (Step 128).

[0016]When it completes normally, it is made for the fork() routine 124 to become parents of the newly created child process layer 106 in which the process layer 102 with the demon application 108 has the demon application 130. The sign in which it is shown from the fork() routine 124 by which process layer those demon applications are performed by the demon applications 108 and 130 is returned. In in the parent process layer 102, it waits to act as a loop back of the demon application 108 in the layer, and to send work further from the port 110 (Step 129).

[0017]In in **** 106, the demon application 114 in the layer sets up the security attributes of the user name 114 specified by the demand.

[0018]Some POSIX functions are called and the right supplementary group, group ID, and user ID of **** 106 are set up. These calls access the data of the requestor-side user in the security database 134. The child demon application 130 issues a getpwnam() call first, and accesses the security database 134, and, specifically, the user ID and group ID corresponding to the user name 114 are judged (Step 132). The application 130 issues a getgroups() call next, and accesses the security database 134, and the supplementary group corresponding to the user name 114 is judged (Step 138).

[0019]Use this information, the child demon application 130 issues a setgroups() call, and the supplementary group of **** 106 is set as the supplementary group corresponding to the user name 114 (Step 140), A setgid() call is issued, the group ID of **** 106 is set as the group ID corresponding to the user name 114 (Step 142), a setuid() call is issued, and the user ID of **** 106 is set as the user ID corresponding to a user name (Step 144).

[0020]If right POSIX security environment is set up as mentioned above for a new user, the demon application 114 will take out an exec() system call, using the specified user task identifier 118 as a parameter (Step 146). By this function, the kernel exec routine 148 reinitializes the address space 106, eliminates a memory location, sets an execution environment to right process security (Step 150), and passes control after that to the new user task program 154 (Step 152).

[0021]In many POSIX conformity systems, such as the IBM OS/390 operating system with OpenEdition expanded function. In order to change the user identity within a user process, there is disadvantage on the serious performance about the excessive call performed to a security database. In this conventional method, the demon cannot use spawn() service of POSIX, therefore must be made to perform a part of demon application according to a child process.

[0022]Drawing 2 is a figure showing the example of the computer systems 200 (physical machinery is included) incorporating this invention, and the relation between various software layers which perform the task demanded for the specified user ID is illustrated. A demon process layer or the address space 202, an operating system (OS), a kernel layer or the address space 204, and a user task (when created) process layer or the address space 206 is included in these layers.

[0023]There is the demon application 208 including the software which supervises the port or the communication line 210 combined with the remote client (not shown) in the demon process layer 202. The demon application 208 accepts the client request 212 which contains the user ID 214, the password 216, and the identifier 218 of the specific task to perform as an input via the port 210 (Step 220).

[0024]After the demon application 208 verifies the user ID 214 and attests the password 216, it prepares new environment so that the new program image which performs the demanded task may be performed. In order to perform this, the demon application 208 is first set as the user name 214 of the demand 212 which processes the environment variable USERNAME (Step 222). Next, the demon application 208 takes out a spawn() system call to the kernel layer 204, and creates a new process (Step 224). A spawn() system call passes the task identification 218 as a parameter, and passes the user name 214 as the environment variable USERNAME. Next, it acts to the head of a routine as a loop back of the demon application 208, and it obtains work further (Step 236).

[0025]By a spawn() system call (Step 224), the kernel spawn routine 226 in the kernel layer 204 gains control. The spawn routine 226 will create a new user task layer or the address space 206 first, if control is gained (Step 228).

[0026]Next, it refers for the spawn routine 226 to the security database 232 in the kernel layer 204, and it obtains a required POSIX permission preset value (Step 230). Next, right security permission is used, Namely, the user ID and group ID of the new address space 206 which were set up equally to the group ID and user ID which were matched with the user name (specified as the environment variable USERNAME) 214 in the database 232 are used, The new address space 206 is initialized directly.

[0027]At the end, the kernel spawn routine 226, The program image 238 corresponding to the user task (passed as a parameter by the demon application 208) 218 specified by the demand 212 is loaded to the address space 206, Control is passed to the program image and a user task is performed (Step 234).

[0028]In order to avoid the improper use by a remote user, it is necessary to restrict the new environment variable USERNAME only to an authorized user. A privilege required to use this function must be equivalent to a setuid() function. (Demon application is usually so like) When operating as a superuser (user ID =0) with the right to access with the unrestricted demon

application 208, the demon application 208 will have required authority.

[0029]As mentioned above, this invention provides the method of creating the security environment for a user task process having no necessity of adding change to daemon process environment, and without the necessity of performing every portion of demon application within a user task process. This invention enables it to replace a conventional fork() function and exec() function with a spawn() function, and the simplicity and performance enhancement with which a spawn() function is intrinsically provided by it are realized.

[0030]Probably, various corrections are known if it is a person skilled in the art. Probably, it will be clear about this invention to be a system of a UNIX base and that it is usable also with other systems in this invention although the context of the POSIX conformity system specifically explained as mentioned above.

[0031]As a conclusion, the following matters are indicated about the composition of this invention.

[0032](1). A demand specifies a user's identity and a server system has an operating kernel. In the server system with which a daemon process supervises said demand from said user who asks for execution of the specified user task, Are the method of performing said task instead of said user using the suitable security environment for said user, and if the (a) daemon process receives said demand from said user, [1] The step which sets up an environment variable according to said identity specified by said demand, [2] The step which takes out a system call to said operating system kernel, and performs said specified user task in a new address space, (b) If said operating system kernel receives said system call from said daemon process, [1] the step which creates an address space new for said specified user task, and [2] -- with the step which creates security environment according to said environment variable for said specified user task. [3] A method containing the step which starts said specified user task in said new address space.

(2) A method given in the above (1), wherein said operating system kernel is a POSIX conformity operating system kernel.

(3) A method given in the above (1), wherein said system call is a spawn() system call.

(4) A method given in the above (1) which said user has a user name and is characterized by said identity containing said user name.

(5) A method given in the above (4), wherein said environment variable is set up equally to said user name.

(6) Said new address space and said user name have the user ID related with it, respectively, A method given in the above (5), wherein said Step (b) [2] which creates said security environment sets up equally to the user ID of the user name specified by said environment variable the user ID of said new user address space.

(7) A method given in the above (6) judging the user ID of said user name by accessing a

security database.
(8) Said new address space and said user name have the group ID related with it, respectively, A method given in the above (5), wherein said Step (b) [2] which creates said security environment sets up the group ID of said new address space equally to the group ID of the user name specified by said environment variable.
(9). A demand specifies a user's identity and a server system has an operating kernel. In said server system with which a daemon process supervises said demand from said user who asks for execution of the specified user task, Are a device which performs said task instead of said user using the suitable security environment for said user, it is related with the (a) aforementioned daemon process, and the receipt of said demand from said user is answered, [1] Set up an environment variable according to said identity specified as said demand, [2] A means to take out a system call to said operating system kernel, and to perform said specified user task in a new address space, (b) It is related with said operating system kernel, and answer the receipt of said system call from said daemon process, [1] Create an address space new for said specified user task, and create security environment according to the [2] aforementioned environment variable for said specified user task, [3] A device containing a means to start said specified user task in said new address space.
(10) A device given in the above (9), wherein said operating system kernel is a POSIX conformity operating system kernel.
(11) A device given in the above (9), wherein said system call is a spawn() system call.
(12) A device given in the above (9) which said user has a user name and is characterized by said identity containing said user name.
(13) A device given in the above (12), wherein said environment variable is set up equally to said user name.
(14) Said new address space and said user name have the user ID related with it, respectively, A device given in the above (13) containing a means by which said means (b) [2] which creates said security environment sets up the user ID of said new address space equally to the user ID of the user name specified by said environment variable.
(15) A device given in the above (14) judging the user ID of said user name by accessing a security database.
(16) Said address space and said user name have the group ID related with it, respectively, A device given in the above (13), wherein said Step (b) [2] which creates said security environment sets up the group ID of said new address space equally to the group ID of the user name specified by said environment variable.
(17). A demand specifies a user's identity and a server system has an operating system kernel. In said server system with which a daemon process supervises said demand from said user who asks for execution of the specified user task, It is the machine-readable program

storage which carries out to concreteness the program which can be executed with the machinery which performs the method step which performs a task instead of said user using the suitable security environment for said user, If the (a) aforementioned daemon process receives said demand from said user, said method step, [1] The step which sets up an environment variable according to said identity specified as said demand, [2] The step which publishes a system call to said operating system, and performs said specified user task in a new address space, (b) The step which creates the address space where it is new for the user task by which [1] aforementioned specification was carried out when said operating system kernel receives said system call from said daemon process, [2] the step which creates security environment according to said environment variable for said specified user task, and [3] -- the program storage containing the step which starts said specified user task in said new address space.

(18) Program storage given in the above (17), wherein said operating system kernel is a POSIX conformity operating system kernel.

(19) Program storage given in the above (17), wherein said system call is a spawn() system call.

(20) Program storage given in the above (17) which said user has user ID and is characterized by said identity containing said user ID.

(21) Program storage given in the above (20), wherein said environment variable is set up equally to said user ID.

(22) Said new address space and said user name have the user ID related with it, respectively, Program storage given in the above (21) containing the step to which said Step (b) [2] which creates said security environment sets the user ID of said new address space equally to the user ID of the user name specified by said environment variable.

(23) Program storage given in the above (22) judging said user ID of said user name by accessing a security database.

(24) Said new address space and said user name have the group ID related with it, respectively, Program storage given in the above (21) containing the step to which said Step (b) [2] which creates said security environment sets the group ID of said new address space equally to the group ID of the user name specified by said environment variable.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1]It is a figure showing the computer systems incorporating the conventional embodiment which creates the security environment for a users request task.
[Drawing 2]It is a figure showing the computer systems which use the nest of this invention which creates the security environment for a users request task.
[Description of Notations]
200 Computer systems
202 Demon process layer
204 Kernel layer
206 User task process layer
208 Demon application
210 Port
214 User name
226 Kernel spawn routine
232 Security database
238 User task

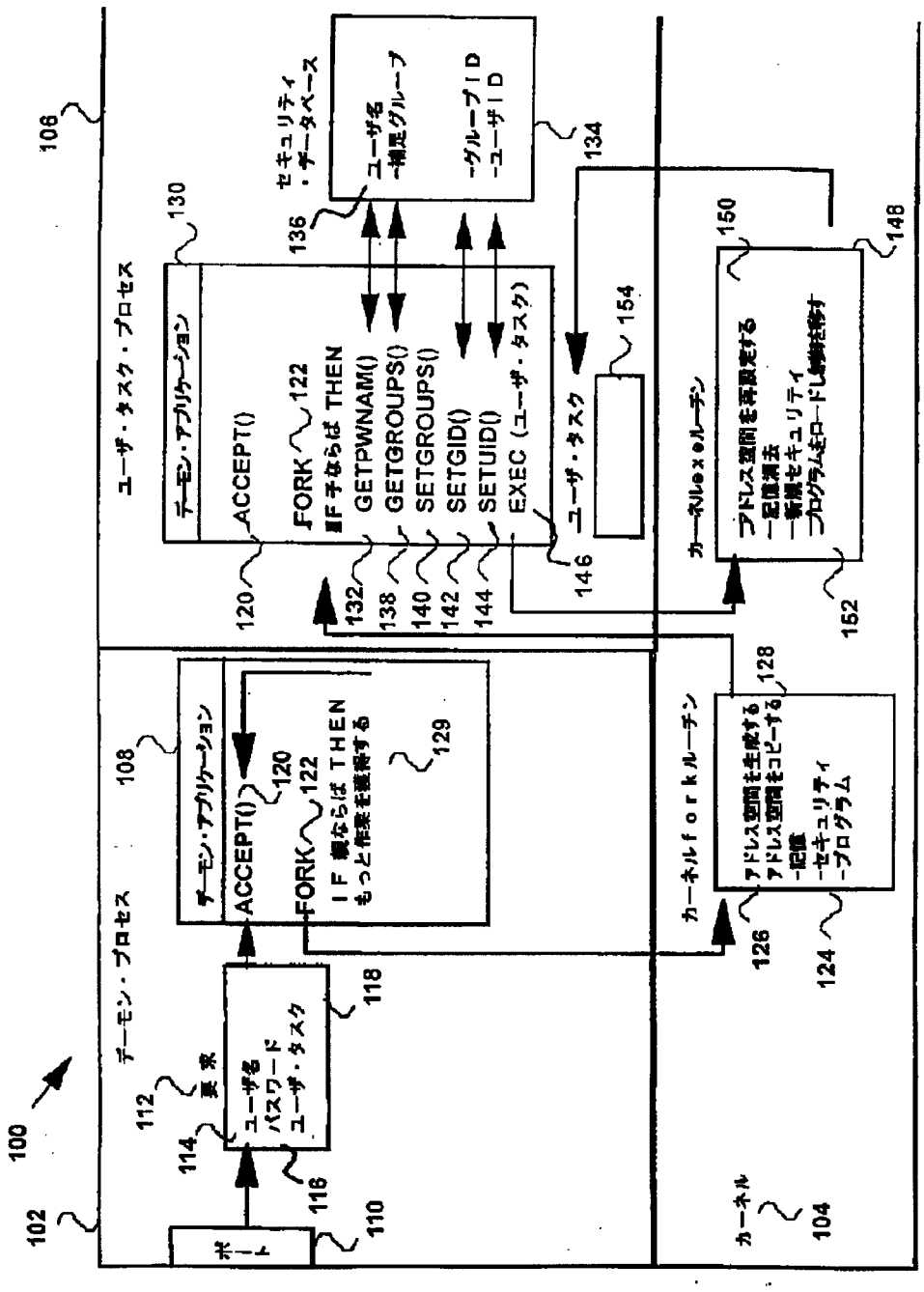
[Translation done.]

* NOTICES *

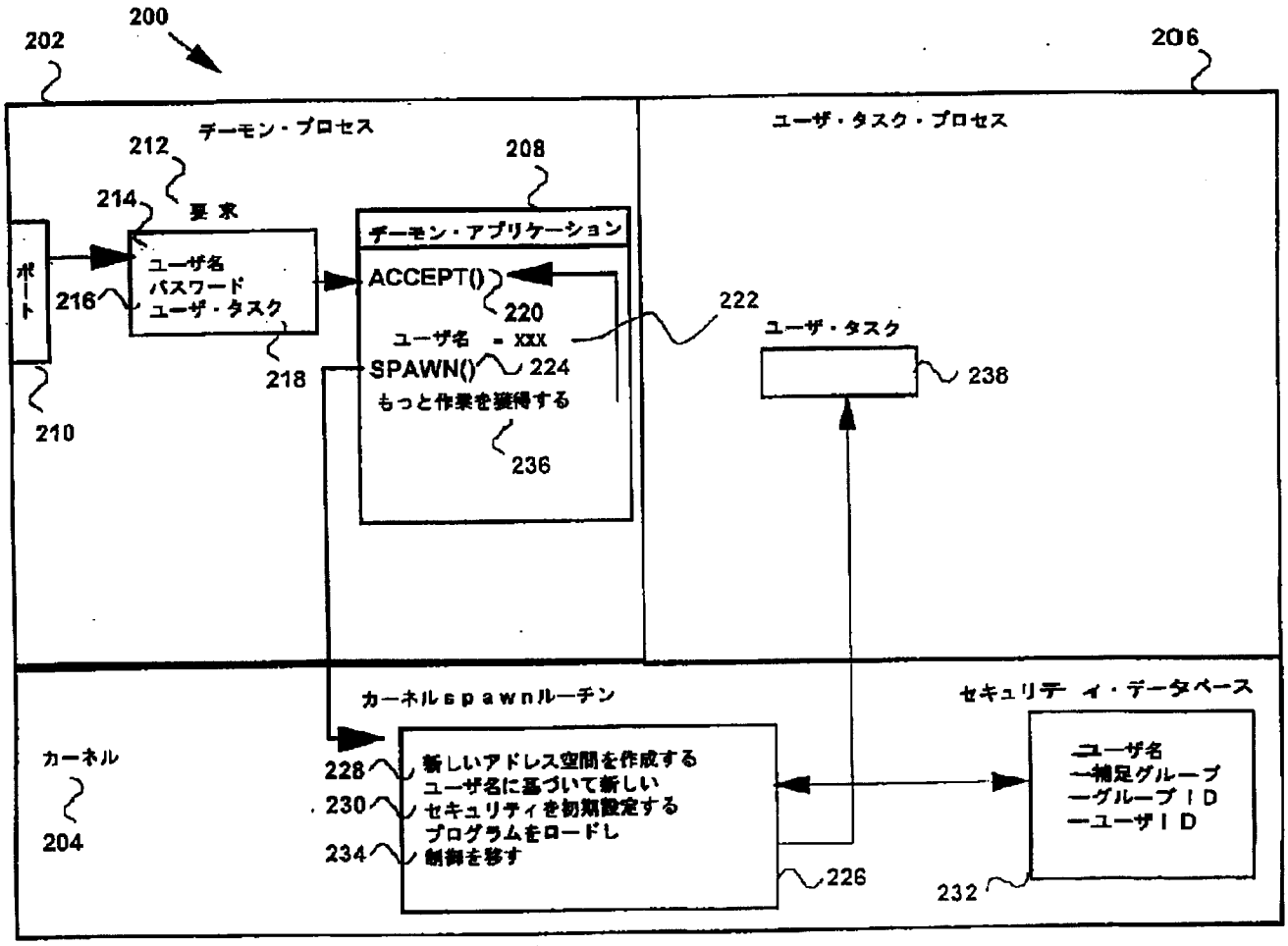
1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DRAWINGS

[Drawing 1]


100
102
ポート
104
カーネル
106
デーモン・プロセス
108
デーモン・アプリケーション
ACCEPT()
IF 親ならば THEN
もっと作業を継続する
120
122
FORK
129
110
112
要求
114
ユーザ名
パスワード
ユーザ・タスク
116
118
カーネルforkルーチン
アドレス空間を生成する
アドレス空間をコピーする
-記憶
-セキュリティ
-プログラム
124
126
128
ユーザ・タスク・プロセス
デーモン・アプリケーション
ACCEPT()
FORK
IF 子ならば THEN
GETPWNAM()
GETGROUPS()
SETGROUPS()
SETGID()
SETUID()
EXEC (ユーザ・タスク)
130
132
138
140
142
144
146
セキュリティ・データベース
ユーザ名
-補足グループ
-グループID
-ユーザID
134
136
154
カーネルexecルーチン
プロセス空間を再設定する
-記憶消去
-新規セキュリティ
プログラムをロードし実行する
148
150
152

[Drawing 2]



[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－161933

(43)公開日　平成10年(1998)６月19日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　12/14	３１０	Ｇ０６Ｆ　12/14	３１０Ａ
9/46	３４０	9/46	３４０Ｆ

審査請求　未請求　請求項の数24　ＯＬ　（全 10 頁）

(21)出願番号　特願平9－254505

(22)出願日　平成９年(1997)９月19日

(31)優先権主張番号　０８／７２１１４５

(32)優先日　1996年９月26日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州アーモンク　（番地なし）

(72)発明者　ドナルド・フレッド・オールト
アメリカ合衆国12538　ニューヨーク州ハイド・パーク　ルーズベルト・ロード　115

(74)代理人　弁理士　坂口　博　（外１名）

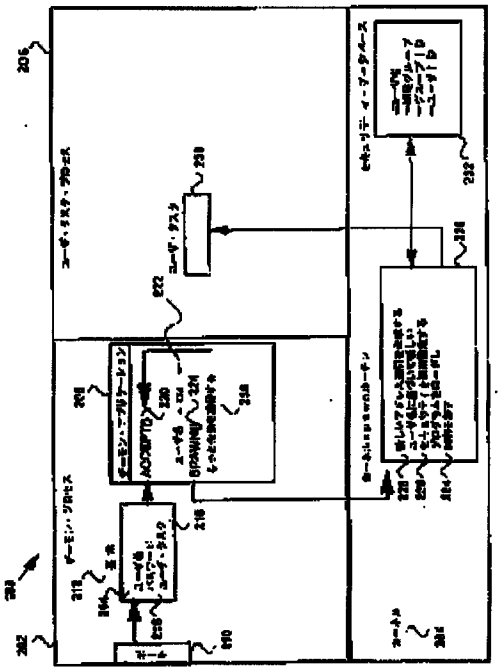
最終頁に続く

(54)【発明の名称】　クライアント／サーバ・システムにおける適切なセキュリティ環境でのユーザ・タスク実行方法及び装置

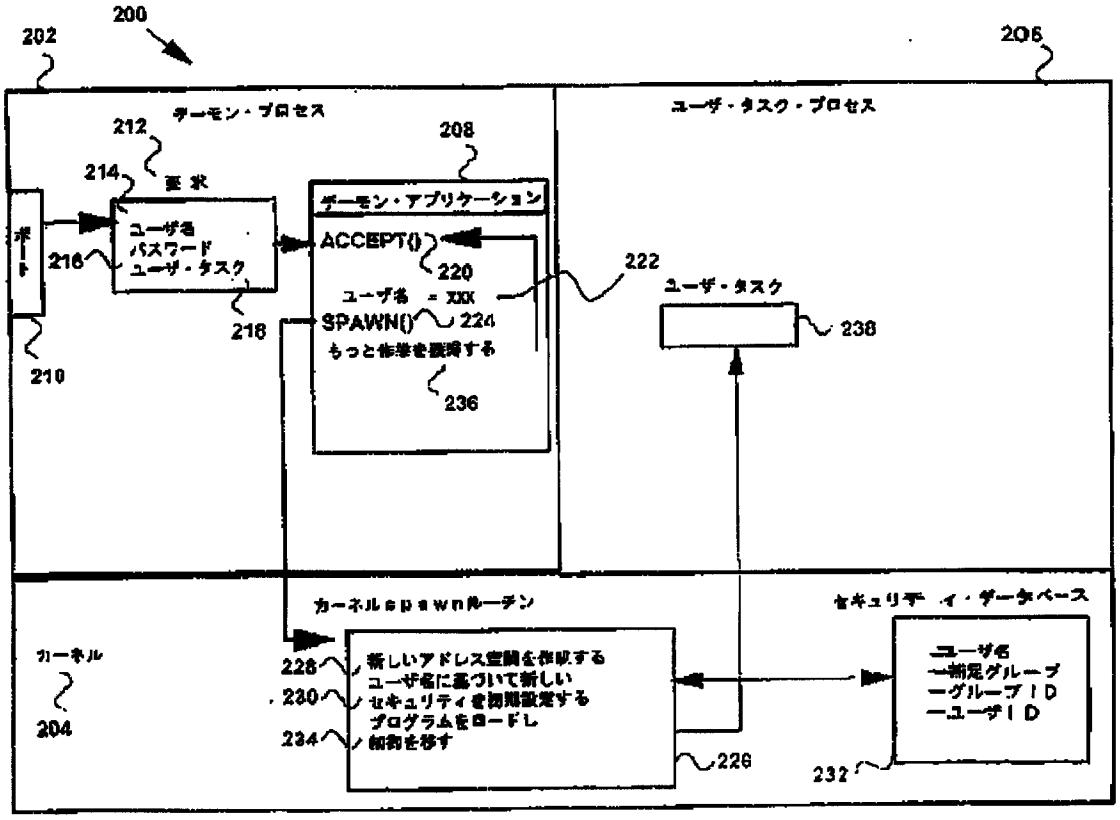
(57)【要約】

【課題】　クライアント／サーバシステムにおいて、監視デーモンが指定されたタスクをユーザに代わって実行することができるようにする方法及び装置を提供する。

【解決手段】　監視デーモンは、ユーザ要求を受け取ると、要求で指定されているユーザ・アイデンティティに従って環境変数を設定し、オペレーティング・システム・カーネルに対してシステム・コールを出して要求で指定されているユーザ・タスクを作成する。オペレーティング・システム・カーネルは、このシステム・コールに応答して指定されたユーザ・タスクのための新しいアドレス空間を作成し、環境変数に従ってユーザ・タスクのためのセキュリティ環境を作成してからユーザ・タスクを新しいアドレス空間で開始する。



【図2】



フロントページの続き

(72)発明者　アーネスト・スコット・ベンダー
アメリカ合衆国12477　ニューヨーク州ソージャーティーズ　パイン・グローブ・スクール・ロード　27

(72)発明者　マイケル・ゲイリー・スピーゲル
アメリカ合衆国10950　ニューヨーク州モンローサンセット・ハイツ　10

1

【特許請求の範囲】

【請求項1】要求がユーザのアイデンティティを指定し、サーバ・システムがオペレーティング・カーネルを有する、指定されたユーザ・タスクの実行を求める前記ユーザからの前記要求をデーモン・プロセスが監視するサーバ・システムにおいて、前記ユーザのための適切なセキュリティ環境を使用して前記ユーザに代わって前記タスクを実行する方法であって、

（a）デーモン・プロセスが、前記ユーザからの前記要求を受け取ると、［1］前記要求で指定された前記アイデンティティに従って環境変数を設定するステップと、［2］前記オペレーティング・システム・カーネルに対してシステム・コールを出して前記指定されたユーザ・タスクを新しいアドレス空間内で実行するステップと、

（b）前記オペレーティング・システム・カーネルが前記デーモン・プロセスから前記システム・コールを受け取ると、［1］前記指定されたユーザ・タスクのために新しいアドレス空間を作成するステップと、［2］前記環境変数に従って前記指定されたユーザ・タスクのためにセキュリティ環境を作成するステップと、［3］前記新しいアドレス空間内で前記指定されたユーザ・タスクを開始するステップとを含む方法。

【請求項2】前記オペレーティング・システム・カーネルがＰＯＳＩＸ準拠オペレーティング・システム・カーネルであることを特徴とする、請求項1に記載の方法。

【請求項3】前記システム・コールがｓｐａｗｎ（）システム・コールであることを特徴とする、請求項1に記載の方法。

【請求項4】前記ユーザがユーザ名を有し、前記アイデンティティが前記ユーザ名を含むことを特徴とする、請求項1に記載の方法。

【請求項5】前記環境変数が前記ユーザ名と等しく設定されることを特徴とする、請求項4に記載の方法。

【請求項6】前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたユーザＩＤを有し、前記セキュリティ環境を作成する前記ステップ（b）［2］が、前記新しいユーザ・アドレス空間のユーザＩＤを、前記環境変数によって指定されたユーザ名のユーザＩＤと等しく設定することを特徴とする、請求項5に記載の方法。

【請求項7】前記ユーザ名のユーザＩＤを、セキュリティ・データベースにアクセスすることによって判断することを特徴とする、請求項6に記載の方法。

【請求項8】前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたグループＩＤを有し、前記セキュリティ環境を作成する前記ステップ（b）［2］が、前記新しいアドレス空間のグループＩＤを、前記環境変数によって指定されたユーザ名のグループＩＤと等しく設定することを特徴とする、請求項5に記載の方法。

2

【請求項9】要求がユーザのアイデンティティを指定し、サーバ・システムがオペレーティング・カーネルを有する、指定されたユーザ・タスクの実行を求める前記ユーザからの前記要求をデーモン・プロセスが監視する前記サーバ・システムにおいて、前記ユーザのための適切なセキュリティ環境を使用して前記ユーザに代わって前記タスクを実行する装置であって、

（a）前記デーモン・プロセスに関連づけられ、前記ユーザからの前記要求の受取りに応答して、［1］前記要求に指定された前記アイデンティティに従って環境変数を設定し、［2］前記オペレーティング・システム・カーネルに対してシステム・コールを出して新しいアドレス空間内で前記指定されたユーザ・タスクを実行する手段と、

（b）前記オペレーティング・システム・カーネルに関連づけられ、前記デーモン・プロセスからの前記システム・コールの受取りに応答して、［1］前記指定されたユーザ・タスクのために新しいアドレス空間を作成し、［2］前記環境変数に従って前記指定されたユーザ・タスクのためにセキュリティ環境を作成し、［3］前記新しいアドレス空間内で前記指定されたユーザ・タスクを開始する手段とを含む装置。

【請求項10】前記オペレーティング・システム・カーネルがＰＯＳＩＸ準拠オペレーティング・システム・カーネルであることを特徴とする、請求項9に記載の装置。

【請求項11】前記システム・コールがｓｐａｗｎ（）システム・コールであることを特徴とする、請求項9に記載の装置。

【請求項12】前記ユーザがユーザ名を有し、前記アイデンティティが前記ユーザ名を含むことを特徴とする、請求項9に記載の装置。

【請求項13】前記環境変数が前記ユーザ名と等しく設定されることを特徴とする、請求項12に記載の装置。

【請求項14】前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたユーザＩＤを有し、前記セキュリティ環境を作成する前記手段（b）［2］が、前記新しいアドレス空間のユーザＩＤを、前記環境変数によって指定されたユーザ名のユーザＩＤに等しく設定する手段を含むことを特徴とする、請求項13に記載の装置。

【請求項15】前記ユーザ名のユーザＩＤを、セキュリティ・データベースにアクセスすることによって判断することを特徴とする、請求項14に記載の装置。

【請求項16】前記アドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたグループＩＤを有し、前記セキュリティ環境を作成する前記ステップ（b）［2］が、前記新しいアドレス空間のグループＩＤを、前記環境変数によって指定されたユーザ名のグループＩＤに等しく設定することを特徴とする、請求項13に記

載の装置。

【請求項17】要求がユーザのアイデンティティを指定し、サーバ・システムがオペレーティング・システム・カーネルを有する、指定されたユーザ・タスクの実行を求める前記ユーザからの前記要求をデーモン・プロセスが監視する前記サーバ・システムにおいて、前記ユーザのための適切なセキュリティ環境を使用して前記ユーザに代わってタスクを実行する方法ステップを行う機械によって実行可能なプログラムを有形に実施する機械可読プログラム記憶装置であって、前記方法ステップが、

（a）前記デーモン・プロセスが、前記ユーザからの前記要求を受け取ると、［1］前記要求に指定された前記アイデンティティに従って環境変数を設定するステップと、［2］前記オペレーティング・システムに対してシステム・コールを発行して前記指定されたユーザ・タスクを新しいアドレス空間内で実行するステップと、

（b）前記オペレーティング・システム・カーネルが、前記デーモン・プロセスから前記システム・コールを受け取ると、［1］前記指定されたユーザ・タスクのために新しいアドレス空間を作成するステップと、［2］前記環境変数に従って前記指定されたユーザ・タスクのためにセキュリティ環境を作成するステップと、［3］前記指定されたユーザ・タスクを前記新しいアドレス空間内で開始するステップとを含む、プログラム記憶装置。

【請求項18】前記オペレーティング・システム・カーネルがＰＯＳＩＸ準拠オペレーティング・システム・カーネルであることを特徴とする、請求項17に記載のプログラム記憶装置。

【請求項19】前記システム・コールがｓｐａｗｎ（）システム・コールであることを特徴とする、請求項17に記載のプログラム記憶装置。

【請求項20】前記ユーザがユーザＩＤを有し、前記アイデンティティが前記ユーザＩＤを含むことを特徴とする、請求項17に記載のプログラム記憶装置。

【請求項21】前記環境変数が前記ユーザＩＤと等しく設定されることを特徴とする、請求項20に記載のプログラム記憶装置。

【請求項22】前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたユーザＩＤを有し、前記セキュリティ環境を作成する前記ステップ（b）［2］が、前記新しいアドレス空間のユーザＩＤを、前記環境変数によって指定されたユーザ名のユーザＩＤと等しく設定するステップを含むことを特徴とする、請求項21に記載のプログラム記憶装置。

【請求項23】前記ユーザ名の前記ユーザＩＤを、セキュリティ・データベースにアクセスすることによって判断することを特徴とする、請求項22に記載のプログラム記憶装置。

【請求項24】前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたグループＩＤを有し、

前記セキュリティ環境を作成する前記ステップ（b）［2］が、前記新しいアドレス空間のグループＩＤを、前記環境変数によって指定されたユーザ名のグループＩＤと等しく設定するステップを含むことを特徴とする、請求項21に記載のプログラム記憶装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、新しいユーザ・アイデンティティと適切な特権を使用してＰＯＳＩＸ環境に新しい作業単位を作成する方法に関する。

【0002】

【従来の技術】（複数の相互接続された機械に作業が分散される）分散コンピューティング・システムは、クライアント／サーバ・モデルに基づいて構築されることが多い。このモデルでは、クライアント・プロセス（または単に「クライアント」）がサーバ・プロセス（または単に「サーバ」）に対して、プリント・サーバの場合はファイルの印刷、ファイル・サーバの場合はファイルの取り出しまたは記憶、あるいはアプリケーション・サーバの場合はアプリケーションの実行など、指定したサービスを実行するように求める要求を出す。クライアント・プロセスとサーバ・プロセスとは同じ物理機械上に存在することができるが、両者は異なる機械上に存在するのが一般的であり、以下の説明においてもそうである。サーバはローカル・エリア・ネットワーク（ＬＡＮ）などのほか、インターネットなどのより広域のネットワークでも使用される。本明細書で特に扱うインターネット・サーバの１つのクラスは、ワールド・ワイド・ウェブにサービスを提供するサーバである。ワールド・ワイド・ウェブは、グラフィカル・コンテンツ（「ウェブ・ページ」）を提供し、ハイパーテキスト転送プロトコル（ＨＴＴＰ）に従ってその他のサービスを行うインターネット・サイトの集まりである。

【0003】ワールド・ワイド・ウェブ（または単に「ウェブ」）上のサーバなどのサーバは、オペレーティング・システムと共に機能し、オペレーティング・システムはサーバが存在する機械上でシステム資源を管理し、基本システム・サービスを行う。これらのオペレーティング・システムは、ＵＮＩＸオペレーティング・システムまたはＵＮＩＸベースのオペレーティング・システムであることが多い。様々なＵＮＩＸオペレーティング・システムの急増により、そのようなシステムによって提供される共通のサービスのセットを定義する様々な努力がなされてきた。このような１つの努力は、１９８８年に初めて発表されたＩＥＥＥ　ＰＯＳＩＸ　１００３．１仕様と、ＩＥＥＥ　ＰＯＳＩＸ　１００３．１ｄなどの補遺である（以下まとめて「ＰＯＳＩＸ」と呼ぶ）。

【0004】ＰＯＳＩＸ準拠システムでは、ユーザによ

るファイルなどへのアクセスは、各プロセス及び各ファイルに、ユーザのアイデンティティを指定するユーザＩＤとグループＩＤを関連づけることによって制御する。各ファイルには、３つの３ビット・フィールドも関連づけられ、各フィールドはｒｗｘビットと呼ばれる３ビットの許可ビットから成る。この３つのフィールドはそれぞれ、ファイルの所有者と、所有者のグループ内の他のユーザと、所有者のグループに入っていないその他のユーザとに対して許可されたアクセスを定義する。各３ビット・フィールド内で、ｒビットはファイルの読み取りができるかどうかを指定し、ｗビットはファイルの書込みができるかどうかを指定し、ｘビットはファイルの実行ができるかどうかを指定する。プロセスがファイルへのアクセスを要求するときは必ず、プロセスとファイルのユーザＩＤとグループＩＤと、そのファイルのために指定された許可ビットを検査して、そのプロセスを実行しているユーザが要求するアクセスを行うことができるかどうかが判断される。このアクセス制御手続きは、当技術分野で周知であり、Ｗ．Ｒ．スティーブンズ（Ｓｔｅｖｅｎｓ）の「ＵＮＩＸ　Ｎｅｔｗｏｒｋ　Ｐｒｏｇｒａｍｍｉｎｇ」（１９９０年）及びＡ．Ｓ．タネンバウム（Ｔａｎｅｎｂａｕｍ）の「Ｍｏｄｅｒｎ　ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ」（１９９２年）などの参考資料に記載されている。

【０００５】サーバ環境でこのようなアクセス制御を実現するために、ＰＯＳＩＸ準拠オペレーティング・システムは、新しいユーザ・アイデンティティを使用して新しい作業単位と、その新しい作業単位が異なる１組の資源にアクセスすることができる新しいセキュリティ環境とを作成することができるサーバの能力をサポートしなければならない。このタイプの作業は、従来、サーバ・システム内で「デーモン」と呼ばれる背景プログラムによって行われてきた。一般に、デーモンは、デーモンの権限のサブセットを有するユーザに代わって要求を処理し、タスクを実行する。システムのセキュリティを維持するために、タスクはそのユーザの権限を使用して新しい作業単位の下で実行されなければならない。これらの新しい作業単位の作成は、ｆｏｒｋ（）、ｓｅｔｕｉｄ（）、ｓｅｔｇｉｄ（）、ｅｘｅｃ（）などの関数及びその他の関連するＰＯＳＩＸサービスを使用して殺われる。

【０００６】デーモンの例は、ＰＯＳＩＸシステム上のウェブ・サーバである。このようなウェブ・サーバは、ポートでユーザがＰＯＳＩＸファイル・システムから文書を取り出す要求を送すのを監視するプログラムである。これを行うために、ウェブ・サーバはまずユーザ名とパスワードを検証しなければならない。次に、ウェブ・サーバはｆｏｒｋ（）関数を使用して新しいアドレス空間を作成し、要求されたユーザ・タスクのために別個の環境を設ける。新しいアドレス空間に入ると、ウェブ・サーバはｇｅｔｇｒｏｕｐｓ（）やｓｅｔｇｒｏｕｐｓ（）などの様々なＰＯＳＩＸ関数を使用して正しいセキュリティ環境を作成し、正しい補足グループＩＤを設定した後、ｓｅｔｇｉｄ（）及びｓｅｔｕｉｄ（）を使用して正しいグループＩＤとユーザＩＤを設定しなければならない。補足グループとグループＩＤとユーザＩＤとは、ＰＯＳＩＸ許可の基礎を形成する。最後にウェブ・サーバはシェルのｅｘｅｃ（）を行い、要求された文書を獲得するシェル・スクリプトを実行する。全体的に見て、これは大量の処理オーバーヘッドを要する長くて複雑なプロセスである。

【０００７】

【発明が解決しようとする課題】ｓｐａｗｎ（）サービスは、ｆｏｒｋ（）サービスとｅｘｅｃ（）サービスを結合して１つのコールにするＰＯＳＩＸ関数である（ＩＥＥＥ　ＰＯＳＩＸ　１００３．１ｄに記載されている）。したがって、ＰＯＳＩＸアプリケーションは、新しいアドレス空間をフォークしてから新しいプログラム・イメージを実行する代わりに、単にその新しいプログラムを異なるアドレス空間に作成して、要求側のアドレス空間をコピーするオーバーヘッドを節約することができる。しかし、この場合の問題は、現在のところ、デーモン自体のセキュリティ環境を変更せずに、新しいプログラム・イメージが制御を獲得する前にユーザ・アイデンティティを変更する方法がないことである。したがって、デーモンが要求された文書を獲得するためにシェル・スクリプトを作成しようとすれば、デーモンのユーザ・アイデンティティの下でＰＯＳＩＸファイル・システムへのアクセスが行われることになり、それによって文書を要求しているユーザのＰＯＳＩＸ許可が置き換えられる可能性がある。ｓｐａｗｎ（）を使用することができないため、デーモン・アプリケーションは前述のように、ユーザ・タスク・プロセス屋で実行する余分の面倒なソフトウェアを備えなければならない。

【０００８】

【課題を解決するための手段】一般には、本発明は、指定したユーザ・タスクを実行するように求めるユーザからの要求をデーモン・プロセスが監視するサーバ・システムにおいて、ユーザのための適切なセキュリティ環境を使用してユーザに代わってタスクを実行する方法及び装置に関する。デーモン・プロセスは、ユーザから要求を受け取ると、その要求で指定されているユーザ・アイデンティティ（たとえばユーザ名）に従って環境変数を設定し、オペレーティング・システム・カーネルに対して、指定されたユーザ・タスクを新しいアドレス空間で実行するシステム・コールを出す。オペレーティング・システム・カーネルは、ＰＯＳＩＸ準拠カーネルとすることができ、その場合、システム・コールはｓｐａｗｎ（）システム・コールとすることができる。オペレーティング・システム・カーネルはデーモン・プロセスから

システム・コールを受け取ると、指定されたユーザ・タスクのために新しいアドレス空間を作成し、指定されたユーザ・タスクのために環境変数に従って新しいセキュリティ環境を作成し、新しいアドレス空間内で指定されたユーザ・タスクを実行する。

【0009】具体的には、本発明によると、新しい環境変数（USERNAME）を作成することによって従来の技術の前述の欠点が解消される。この環境変数が指定されると、spawn()関数によって呼び出されるプログラムが正しいPOSIX許可を使用して制御を獲得する。これを実現するために、spawn()関数を変更して、spawn()関数がこの新しい環境変数を認識し、有効なユーザ名であることを検証した後、指定されたユーザ名のために新しいアドレス空間の初期作成が行われるようにする。新しいユーザ・アイデンティティを使用して実行されると、セキュリティ・データ・ベースに入っているそのユーザのエンティティから残りのPOSIX許可を入手する。新しいアドレス空間とタスクの初期設定が完了すると、spawnルーチンはその新しいプログラム・イメージを現行セキュリティ環境で開始する。

【0010】本発明は、デーモン・プロセス環境に変更を加える必要なしに、ユーザ・タスク・プロセスのためのセキュリティ環境を作成する方法を提供する。さらに、デーモン・アプリケーションのどの部分もユーザ・タスク・プロセスで実行する必要がない（もしそうであればユーザ・タスク・プログラム・イメージに制御を渡す前にユーザ・タスク・プロセス許可の変更を必要とすることになる）。また、本発明は、従来のfork()とexec()の使用をspawn()関数の使用で置き換えることができ、それによってspawn()関数が本質的にもたらす簡略化とパフォーマンス向上を実現する。

【0011】

【発明の実施の形態】図1は、従来の実施態様のデーモンを組み込んだコンピュータ・システム100（物理機械を含む）の例であり、特定のユーザのために要求されたタスクを実行する様々なソフトウェア層間の関係が示されている。これらの層には、デーモン・プロセス層またはアドレス空間102と、オペレーティング・システム（OS）またはカーネル層またはアドレス空間104と、後述のようにして作成されるユーザ・タスク・プロセス層またはアドレス空間106とが含まれる。物理機械は、たとえばS/390並列エンタープライズ・サーバなどのIBM S/390プロセッサとすることができ、カーネル層104はPOSIX準拠OpenEdition構成要素を有するIBM OS/390オペレーティング・システムとすることができる。

【0012】デーモン・プロセス層102には、ユーザに代わって動作して遠隔クライアント（図示せず）に結合されたポートまたは通信回線110を監視するソフトウェアを含むデーモン・アプリケーション108がある。デーモン・アプリケーション108は、入力としてユーザ名114とパスワード116とタスクを指定する識別子118とを含む要求112をポート110を介して受け入れる（ステップ120）。

【0013】ユーザ名（またはログイン名）114は、ユーザに固有に関連づけられ、ユーザが自分自身をシステム100に識別させるために使用する英数字文字列である。システム100に付随するセキュリティ・データベース134内の各ユーザ名114のレコード136に、ユーザ名に固有に関連づけられた整数ユーザID、ユーザ名に固有に関連づけられた整数グループID、ユーザ名に関連づけられた補足グループのリストのほか、ユーザのパスワード及びその他の関係情報が記憶される。ユーザを認証するために、デーモン・アプリケーション108はセキュリティ・データベース134内の指定されたユーザ名114に対応するレコード136にアクセスし、そのレコードに入っているパスワードを調べる。指定されたユーザ名114のレコード136があり、レコード内のパスワードが要求に添付されたパスワード118と一致する場合、要求者はその要求者が自称しているユーザであると認証される。

【0014】デーモン・アプリケーション108はユーザ名114を検証し、パスワード116を認証した後、要求されたタスクを実行する新しいプログラム・イメージを実行するように新しい環境を準備しなければならない。そのために、デーモン・アプリケーション108はカーネル層104にfork()システム・コールを出して新しいプロセスを作成する（ステップ122）。

【0015】fork()システム・コールは、カーネル層104でカーネルforkルーチン124をトリガして制御を獲得させる。forkルーチン124は制御を獲得すると新しいアドレス空間106を作成して初期設定し（ステップ126）、デーモン層102からユーザ・タスク・プロセス層106に記憶属性、セキュリティ属性、及びプロセス実行属性をコピーする（ステップ128）。

【0016】正常に完了すると、fork()ルーチン124はデーモン・アプリケーション108を持つプロセス層102が、デーモン・アプリケーション130を持つ新たに作成された子プロセス層106の親になるようにする。fork()ルーチン124から、デーモン・アプリケーション108及び130にそれらのデーモン・アプリケーションがどのプロセス層で実行されているかを示す標識を返す。親プロセス層102内の場合、その層内のデーモン・アプリケーション108はループ・バックして、ポート110からさらに作業が送られてくるのを待つ（ステップ129）。

【0017】子層106内の場合、その層内のデーモン

・アプリケーション１１４は要求で指定されたユーザ名１１４のセキュリティ属性を設定する。

【００１８】いくつかのＰＯＳＩＸ関数が呼び出され、子層１０６の正しい補足グループとグループＩＤとユーザＩＤとが設定される。これらのコールは、セキュリティ・データベース１３４内の要求側ユーザのデータにアクセスする。具体的には、子デーモン・アプリケーション１３０はまずｇｅｔｐｗｎａｍ（）コールを出してセキュリティ・データベース１３４にアクセスし、ユーザ名１１４に対応するユーザＩＤとグループＩＤを判断する（ステップ１３２）。アプリケーション１３０は次にｇｅｔｇｒｏｕｐｓ（）コールを出してセキュリティ・データベース１３４にアクセスし、ユーザ名１１４に対応する補足グループを判断する（ステップ１３８）。

【００１９】この情報を使用して、子デーモン・アプリケーション１３０はｓｅｔｇｒｏｕｐｓ（）コールを出して子層１０６の補足グループをユーザ名１１４に対応する補足グループに設定し（ステップ１４０）、ｓｅｔｇｉｄ（）コールを出して子層１０６のグループＩＤをユーザ名１１４に対応するグループＩＤに設定し（ステップ１４２）、ｓｅｔｕｉｄ（）コールを出して子層１０６のユーザＩＤをユーザ名に対応するユーザＩＤに設定する（ステップ１４４）。

【００２０】上述のようにして新しいユーザのために正しいＰＯＳＩＸセキュリティ環境が設定されると、デーモン・アプリケーション１１４は指定されたユーザ・タスク識別子１１８をパラメータとして使用してｅｘｅｃ（）システム・コールを出す（ステップ１４６）。この関数によってカーネルｅｘｅｃルーチン１４８はアドレス空間１０６を初期設定し直し、記憶域を消去し、正しいプロセス・セキュリティと実行環境を設定し（ステップ１５０）、その後で新しいユーザ・タスク・プログラム１５４に制御を渡す（ステップ１５２）。

【００２１】ＯｐｅｎＥｄｉｔｉｏｎ拡張機能付きのＩＢＭ　ＯＳ／３９０オペレーティング・システムなどの多くのＰＯＳＩＸ準拠システムでは、ユーザ・プロセス内のユーザ・アイデンティティを変更するためにセキュリティ・データベースに対して行われる過度のコールに関する重大なパフォーマンス上の不利がある。また、この従来の方法では、デーモンはＰＯＳＩＸのｓｐａｗｎ（）サービスを利用することができず、したがってデーモン・アプリケーションの一部を子プロセスで実行するようにしなければならない。

【００２２】図２は、本発明を組み込んだコンピュータ・システム２００（物理機械を含む）の例を示す図であり、指定されたユーザＩＤのために要求されたタスクを実行する様々なソフトウェア層間の関係が図示されている。これらの層には、デーモン・プロセス層またはアドレス空間２０２と、オペレーティング・システム（ＯＳ）またはカーネル層またはアドレス空間２０４と、（作成された場合は）ユーザ・タスク・プロセス層またはアドレス空間２０６とが含まれる。

【００２３】デーモン・プロセス層２０２には、遠隔クライアント（図示せず）に結合されたポートまたは通信回線２１０を監視するソフトウェアを含むデーモン・アプリケーション２０８がある。デーモン・アプリケーション２０８は、入力としてユーザＩＤ２１４とパスワード２１６と実行する特定のタスクの識別子２１８とを含むクライアント要求２１２をポート２１０を介して受け入れる（ステップ２２０）。

【００２４】デーモン・アプリケーション２０８は、ユーザＩＤ２１４を検証し、パスワード２１６を認証した後、要求されたタスクを行う新しいプログラム・イメージを実行するように新しい環境を準備する。これを行うために、デーモン・アプリケーション２０８はまず、環境変数ＵＳＥＲＮＡＭＥを処理する要求２１２のユーザ名２１４に設定する（ステップ２２２）。デーモン・アプリケーション２０８は次に、カーネル層２０４に対してｓｐａｗｎ（）システム・コールを出して新しいプロセスを作成する（ステップ２２４）。ｓｐａｗｎ（）システム・コールはパラメータとしてタスク識別子２１８を渡し、環境変数ＵＳＥＲＮＡＭＥとしてユーザ名２１４を渡す。次にデーモン・アプリケーション２０８はルーチンの先頭にループ・バックしてさらに作業を入手する（ステップ２３６）。

【００２５】ｓｐａｗｎ（）システム・コール（ステップ２２４）によって、カーネル層２０４内のカーネルｓｐａｗｎルーチン２２６が制御を獲得する。ｓｐａｗｎルーチン２２６は制御を獲得するとまず新しいユーザ・タスク層またはアドレス空間２０６を作成する（ステップ２２８）。

【００２６】ｓｐａｗｎルーチン２２６は次に、カーネル層２０４内のセキュリティ・データベース２３２に照会して必要なＰＯＳＩＸ許可設定値を入手する（ステップ２３０）。次に、正しいセキュリティ許可を使用して、すなわちデータベース２３２内の（環境変数ＵＳＥＲＮＡＭＥに指定されている）ユーザ名２１４に対応づけられたグループＩＤとユーザＩＤに等しく設定された新しいアドレス空間２０６のユーザＩＤとグループＩＤを使用して、新しいアドレス空間２０６を直接初期設定する。

【００２７】最後に、カーネルｓｐａｗｎルーチン２２６は、要求２１２で指定された（デーモン・アプリケーション２０８によってパラメータとして渡された）ユーザ・タスク２１８に対応するプログラム・イメージ２３８をアドレス空間２０６にロードし、そのプログラム・イメージに制御を渡してユーザ・タスクを実行する（ステップ２３４）。

【００２８】遠隔ユーザによる悪用を避けるために、新しい環境変数ＵＳＥＲＮＡＭＥは許可されたユーザのみ

に制限する必要がある。この機能を使用するのに必要な特権は、ｓｅｔｕｉｄ（）関数と同等でなければならない。（デーモン・アプリケーションは通常そうであるように）デーモン・アプリケーション２０８が無制限のアクセス権を持つスーパーユーザ（ユーザＩＤ＝０）として動作する場合、デーモン・アプリケーション２０８は必要な権限を持つことになる。

【００２９】上述のように、本発明は、デーモン・プロセス環境に変更を加える必要なしに、また、デーモン・アプリケーションのどの部分もユーザ・タスク・プロセス内で実行する必要なしに、ユーザ・タスク・プロセスのためのセキュリティ環境を作成する方法を提供する。また、本発明は、従来のｆｏｒｋ（）関数とｅｘｅｃ（）関数をｓｐａｗｎ（）関数で置き換えることができるようにし、それによってｓｐａｗｎ（）関数が本質的に備える簡略さとパフォーマンス強化を実現する。

【００３０】当業者なら様々な修正がわかるであろう。上記のように本発明についてＵＮＩＸベースのシステム、具体的にはＰＯＳＩＸ準拠システムの文脈で説明したが、本発明はその他のシステムでも使用可能であることは明らかであろう。

【００３１】まとめとして、本発明の構成に関して以下の事項を開示する。

【００３２】（１）要求がユーザのアイデンティティを指定し、サーバ・システムがオペレーティング・カーネルを有する、指定されたユーザ・タスクの実行を求める前記ユーザからの前記要求をデーモン・プロセスが監視するサーバ・システムにおいて、前記ユーザのための適切なセキュリティ環境を使用して前記ユーザに代わって前記タスクを実行する方法であって、

（ａ）デーモン・プロセスが、前記ユーザからの前記要求を受け取ると、［１］前記要求で指定された前記アイデンティティに従って環境変数を設定するステップと、［２］前記オペレーティング・システム・カーネルに対してシステム・コールを出して前記指定されたユーザ・タスクを新しいアドレス空間内で実行するステップと、

（ｂ）前記オペレーティング・システム・カーネルが前記デーモン・プロセスから前記システム・コールを受け取ると、［１］前記指定されたユーザ・タスクのために新しいアドレス空間を作成するステップと、［２］前記環境変数に従って前記指定されたユーザ・タスクのためにセキュリティ環境を作成するステップと、［３］前記新しいアドレス空間内で前記指定されたユーザ・タスクを開始するステップとを含む方法。

（２）前記オペレーティング・システム・カーネルがＰＯＳＩＸ準拠オペレーティング・システム・カーネルであることを特徴とする、上記（１）に記載の方法。

（３）前記システム・コールがｓｐａｗｎ（）システム・コールであることを特徴とする、上記（１）に記載の方法。

（４）前記ユーザがユーザ名を有し、前記アイデンティティが前記ユーザ名を含むことを特徴とする、上記（１）に記載の方法。

（５）前記環境変数が前記ユーザ名と等しく設定されることを特徴とする、上記（４）に記載の方法。

（６）前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたユーザＩＤを有し、前記セキュリティ環境を作成する前記ステップ（ｂ）［２］が、前記新しいユーザ・アドレス空間のユーザＩＤを、前記環境変数によって指定されたユーザ名のユーザＩＤと等しく設定することを特徴とする、上記（５）に記載の方法。

（７）前記ユーザ名のユーザＩＤを、セキュリティ・データベースにアクセスすることによって判断することを特徴とする、上記（６）に記載の方法。

（８）前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたグループＩＤを有し、前記セキュリティ環境を作成する前記ステップ（ｂ）［２］が、前記新しいアドレス空間のグループＩＤを、前記環境変数によって指定されたユーザ名のグループＩＤと等しく設定することを特徴とする、上記（５）に記載の方法。

（９）要求がユーザのアイデンティティを指定し、サーバ・システムがオペレーティング・カーネルを有する、指定されたユーザ・タスクの実行を求める前記ユーザからの前記要求をデーモン・プロセスが監視する前記サーバ・システムにおいて、前記ユーザのための適切なセキュリティ環境を使用して前記ユーザに代わって前記タスクを実行する装置であって、

（ａ）前記デーモン・プロセスに関連づけられ、前記ユーザからの前記要求の受取りに応答して、［１］前記要求に指定された前記アイデンティティに従って環境変数を設定し、［２］前記オペレーティング・システム・カーネルに対してシステム・コールを出して新しいアドレス空間内で前記指定されたユーザ・タスクを実行する手段と、

（ｂ）前記オペレーティング・システム・カーネルに関連づけられ、前記デーモン・プロセスからの前記システム・コールの受取りに応答して、［１］前記指定されたユーザ・タスクのために新しいアドレス空間を作成し、［２］前記環境変数に従って前記指定されたユーザ・タスクのためにセキュリティ環境を作成し、［３］前記新しいアドレス空間内で前記指定されたユーザ・タスクを開始する手段とを含む装置。

（１０）前記オペレーティング・システム・カーネルがＰＯＳＩＸ準拠オペレーティング・システム・カーネルであることを特徴とする、上記（９）に記載の装置。

（１１）前記システム・コールがｓｐａｗｎ（）システム・コールであることを特徴とする、上記（９）に記載の装置。

（12）前記ユーザがユーザ名を有し．前記アイデンティティが前記ユーザ名を含むことを特徴とする．上記（9）に記載の装置。

（13）前記環境変数が前記ユーザ名と等しく設定されることを特徴とする，上記（12）に記載の装置。

（14）前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたユーザＩＤを有し．前記セキュリティ環境を作成する前記手段（b）［2］が，前記新しいアドレス空間のユーザＩＤを．前記環境変数によって指定されたユーザ名のユーザＩＤに等しく設定する手段を含むことを特徴とする，上記（13）に記載の装置。

（15）前記ユーザ名のユーザＩＤを．セキュリティ・データベースにアクセスすることによって判断することを特徴とする．上記（14）に記載の装置。

（16）前記アドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたグループＩＤを有し，前記セキュリティ環境を作成する前記ステップ（b）［2］が，前記新しいアドレス空間のグループＩＤを，前記環境変数によって指定されたユーザ名のグループＩＤに等しく設定することを特徴とする．上記（13）に記載の装置。

（17）要求がユーザのアイデンティティを指定し，サーバ・システムがオペレーティング・システム・カーネルを有する，指定されたユーザ・タスクの実行を求める前記ユーザからの前記要求をデーモン・プロセスが監視する前記サーバ・システムにおいて．前記ユーザのための適切なセキュリティ環境を使用して前記ユーザに代わってタスクを実行する方法ステップを行う機械によって実行可能なプログラムを有形に実施する機械可読プログラム記憶装置であって、前記方法ステップが，

（a）前記デーモン・プロセスが，前記ユーザからの前記要求を受け取ると，［1］前記要求に指定された前記アイデンティティに従って環境変数を設定するステップと．［2］前記オペレーティング・システムに対してシステム・コールを発行して前記指定されたユーザ・タスクを新しいアドレス空間内で実行するステップと．

（b）前記オペレーティング・システム・カーネルが，前記デーモン・プロセスから前記システム・コールを受け取ると，［1］前記指定されたユーザ・タスクのために新しいアドレス空間を作成するステップと，［2］前記環境変数に従って前記指定されたユーザ・タスクのためにセキュリティ環境を作成するステップと，［3］前記指定されたユーザ・タスクを前記新しいアドレス空間内で開始するステップとを含む、プログラム記憶装置。

（18）前記オペレーティング・システム・カーネルがＰＯＳＩＸ準拠オペレーティング・システム・カーネルであることを特徴とする．上記（17）に記載のプログラム記憶装置。

（19）前記システム・コールがｓｐａｗｎ（）システム・コールであることを特徴とする．上記（17）に記載のプログラム記憶装置。

（20）前記ユーザがユーザＩＤを有し，前記アイデンティティが前記ユーザＩＤを含むことを特徴とする，上記（17）に記載のプログラム記憶装置。

（21）前記環境変数が前記ユーザＩＤと等しく設定されることを特徴とする，上記（20）に記載のプログラム記憶装置。

（22）前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたユーザＩＤを有し．前記セキュリティ環境を作成する前記ステップ（b）［2］が．前記新しいアドレス空間のユーザＩＤを，前記環境変数によって指定されたユーザ名のユーザＩＤと等しく設定するステップを含むことを特徴とする，上記（21）に記載のプログラム記憶装置。

（23）前記ユーザ名の前記ユーザＩＤを，セキュリティ・データベースにアクセスすることによって判断することを特徴とする，上記（22）に記載のプログラム記憶装置。

（24）前記新しいアドレス空間と前記ユーザ名とがそれぞれそれに関連づけられたグループＩＤを有し．前記セキュリティ環境を作成する前記ステップ（b）［2］が．前記新しいアドレス空間のグループＩＤを．前記環境変数によって指定されたユーザ名のグループＩＤと等しく設定するステップを含むことを特徴とする．上記（21）に記載のプログラム記憶装置。

【図面の簡単な説明】

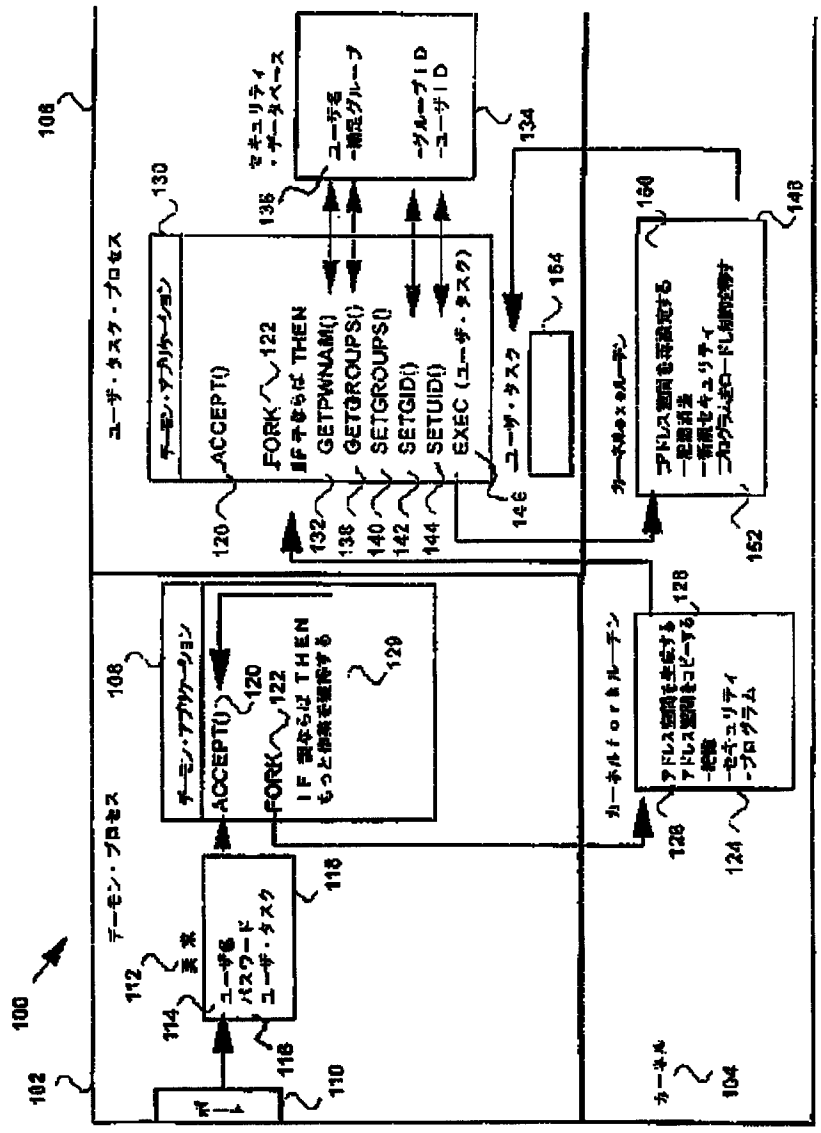
【図1】ユーザ要求タスクのためのセキュリティ環境を作成する従来の実施態様を組み込んだコンピュータ・システムを示す図である。

【図2】ユーザ要求タスクのためのセキュリティ環境を作成する本発明の組込みを使用したコンピュータ・システムを示す図である。

【符号の説明】

200　コンピュータ・システム
202　デーモン・プロセス層
204　カーネル層
206　ユーザ・タスク・プロセス層
208　デーモン・アプリケーション
210　ポート
214　ユーザ名
226　カーネルｓｐａｗｎルーチン
232　セキュリティ・データベース
238　ユーザ・タスク

【図1】


100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
129
130
132
134
136
138
140
142
144
146
148
150
152
154
156
デーモン・プロセス
ユーザ・タスク・プロセス
カーネル
ポート
要求
ユーザ名
パスワード
ユーザ・タスク
デーモン・アプリケーション
ACCEPT()
FORK
IF 真ならば THEN
もっと仕事を続ける
カーネルルーチン
アドレス空間を生成する
アドレス空間をコピーする
-処理
-セキュリティ
-プログラム
ユーザ・タスク
EXEC (ユーザ・タスク)
SETUID()
SETGID()
SETGROUPS()
GETGROUPS()
GETPWNAM()
IF 真ならば THEN
セキュリティ・データベース
ユーザを
-属性グループ
-グループID
-ユーザID
アドレス空間を解放する
-処理終了
-新規セキュリティ
プログラムをロードし始める